WEST

Generate Collection Print

L4: Entry 8 of 15 File: USPT Sep 8, 1998

DOCUMENT-IDENTIFIER: US 5806072 A
TITLE: Electronic imaging apparatus having hierarchical image data storage structure for computer-compatible image data management

Brief Summary Text (5):
Meanwhile, the personal computer DOS (Disk Operating System) currently popular in various fields usually makes use of hierarchical directories, i.e., directories in a hierarchical organization for managing data files to meet a demand for quick retrieval or selection of many databases or programs. In a method of managing data files with hierarchical directories, the location of the file data recording area in the memory and the name of each data file are registered in a register or commonly-named directory. These files are classified as desired into a tree-like hierarchy with hierarchical directories according to their kind or content, and the desired data files are retrieved by designating the corresponding directory names and file names.

Brief Summary Text (6):
When it is desired to observe the status of data file classification with hierarchical directories, a hierarchical directory tree is displayed on a monitor, whereby the operator of the personal computer can know the location of data file registration. FIG. 20 shows an example of the tree display. Mark ".Yen." designates a root directory which branches into three subdirectories named "DOS", "BAT" and "USR". The subdirectory "USR" branches into low-order level subdirectories "C" and "PASCAL". The operator can easily confirm an intended data file designate by operating a keyboard while observing the directory tree display. Such method of file designation and selection is very useful, particularly in the case of a file structure which has many complicated branches.

Brief Summary Text (19):
When an object is successively picked-up with the SMC having an IC memory card of the above storage format, the file names in the root directories in the memory management area are called in the order of the pick-up. The actual image data, however, are stored randomly in the vacant data file areas searched by a system controller to be described later among areas as a plurality of clusters, and the start cluster is stored in a root directory. In other words, while the directory entry of the file names and the start clusters are produced in the order of the pick-up, the actual image data of each picked-up frame is stored in a random location. Further, with the card, in which the order of the pick-up and the order of the storage is made out of correspondence by copying or the like, the order of reproduction is determined according to the order of the entry in the root directory.

Brief Summary Text (25):
In the case of using a memory card with both of the SMC and the personal computer, it is necessary to work out the format of the memory card used in the SMC to conform to the personal computer DOS (disk operating system). At present, JEIDA Ver. 4.1 noted above serves as a memory card standard, and standardized memory cards are used with the SMC. FIG. 46 shows a logical format of a memory card for the SMC, conforming to the above standardized memory card capable of use in a personal computer as well. This format is the same as the personal computer DOS format. As is shown, a first portion of the memory is a boot sector area 51, in which management area size information and parameters for file management are stored. Following the boot sector area 51 are a FAT (file allocation table) 52, a root directory area 53 and then a data area 54, in which image data are stored. In the FAT area 52 is written a FAT entry, i.e., the information about the succession of locations of the file data storage. The location of the data stored in the memory is represented by a cluster number which serves as a data write unit number. In the root directory area 53 is written a directory entry, i.e., information indicative of the registration of hierarchically highest-order level directories among files registered after classification with a hierarchical

organization. In the data area 54, the picked-up image data are written in the form of files. In addition, when a subdirectory is formed in a directory, its directory entry is also written. In a personal computer, the data area 54 corresponds to the file data area.

Drawing Description Text (5):
FIG. 4 shows a tree structure of the root directory of the memory card of an SMC according to the present invention;

Drawing Description Text (6):
FIG. 5 shows tree structures of the root directory and the subdirectory of an SMC according to the present invention;

Drawing Description Text (7):
FIG. 6 shows drawings for explaining the registration of the root directory and subdirectory of an SMC according to the present invention;

Drawing Description Text (19):
FIG. 18 shows a flow chart of the subroutine "root directory routine" called by the "directory UP routine" of FIG. 16;

Drawing Description Text (31):
FIG. 30 shows the root directory and entry format in the memory control region of the IC memory card;

Drawing Description Text (32):
FIG. 31 shows the root directory and data file region in the memory control region of the IC memory card;

Drawing Description Text (141):
FIG. 142 shows a structure of the root directory of FIG. 141;

Detailed Description Text (9):
As image data of the SMC, a frame of the image data is dealt with as a file as noted above. The recording and play of the image files are managed by utilizing the hierarchical directory file system. The file system of the SMC will now be described briefly. In the memory card 14, memory areas are allocated as a set of sections, i.e., a FAT section, in which the FAT information noted above is stored, a root directory section, in which basic directory information is stored, and a data section, in which the image data is stored. When storing image data in an out-of-use state, i.e., in a state without any new directory area provided, file name, file A, file B, . . . are registered in a root directory as shown in FIG. 4,and the image data are stored successively in designated memory areas. In FIG. 4, mark .Yen. designates a root directory. When a subdirectory b is produced in the root directory as shown in FIG. 5, it is registered in the root directory, and a second hierarchical level subdirectory of directory name b is secured under the root directory. In the subdirectory b, the image data files with file names of E to G can be registered. As their directory information, cluster Nos. indicative of the positions of memory areas with the file data stored therein are written in a directory entry to be described later, and files are accessed according to this information.

Detailed Description Text (10):
In the same subdirectory, a third hierarchical level subdirectory d can be produced. In the subdirectory d, file J, file K, . . . can be stored. Further, as shown in FIG. 5, a separate subdirectory c may be produced as a second hierarchical level subdirectory in the root directory .Yen. to register file H, file I, . . . in this subdirectory. In the above subdirectories may be stored image files concerning pictures of scenes belonging to the same theme or those classified for pick-up periods or as to whether pictures are advancement or retreat storage pictures.

Detailed Description Text (11):
When reproducing the image data, it may be desired not only to successively reproduce files A to C of the image data in the root directory but also to reproduce only the classified image data files E, F and G. In such a case, the subdirectory c can be selected for reproduction. By so doing, an intended image can be reproduced quickly.

Detailed Description Text (12):
FIG. 6 shows a specific example of the hierarchical directory structure of FIG. 5 for SMC image data files. In this hierarchical directory structure, a root directory a,

second hierarchical level subdirectories b and c and a third hierarchical level subdirectory d are organized in a tree-like relation to one another. In a first portion of each directory information storage area except for the root directory, a 2-byte directory pointer indicative of the high-order level directory information storage address is written. This pointer permits returning to the high-order level directory position. In the following area, the directory entry is stored as the subdirectory or image data file directory information.

Detailed Description Text (14):
When registering the image data file to be stored in the subdirectory by using the hierarchical directory structure noted above, a directory is produced by operating the MD switch 11f in the set of switches 11 shown in FIG. 3 to secure an image file storage area. Then, to shift up or down the directory position for file accessing the hierarchical directory retrieval position is shifted up or down by operating the D-UP switch 11d or the D-DOWN switch 11e. Then, the data file to be stored is selected by operating the UP and DOWN switches 11d and 11c. A tree traversal system access can be obtained by continuously operating the UP and DOWN switches 11b and 11c. More specifically, when the subdirectory is accessed during successive accessing of files in the directory, files in the pertinent low-order level hierarchical subdirectory are now accessed. When the subdirectory file accessing is over, now high-order level directory files are accessed.

Detailed Description Text (18):
FIG. 9 shows states the display 12 when the storage file is shifted by operating the UP and DOWN switches 11b and 11c in the recording mode. In the display shown in 9(A), "01" is displayed on the first display section 12a, indicating that the image file to be stored is a file of file name 1 in the root directory (which corresponds to file A in FIG. 6). In the display shown in FIG. 9(B), "02" is displayed on the first display section 12a, indicating that the image file to be stored is of file name 2 in the root directory (which corresponds to file B in FIG. 6). It is to be appreciated that the displayed file name may correspond to the frame No., which is a feature of this embodiment. In the display shown in FIG. 9(C), "04" and "01" are displayed in the respective first and second display sections 12a and 12b, indicating that the designated image file to be stored is of file name 1 (corresponding to file E in FIG. 6) in a subdirectory designated by directory name 4 in the root directory (corresponding to directory b in FIG. 6). The display shown in FIG. 9(D) indicates that the designated image file to be stored is a file of the file name 2 (corresponding to file F in FIG. 6) in a subdirectory designated by directory name 4 in the root directory (corresponding to directory b in FIG. 6). The switching of the files shown in FIGS. 9(A) to 9(D) is effected by operating the UP and DOWN switches 11b and 11c in the set of switches 11. When the UP switch 11b is further held depressed, the directory information shown in FIG. 6 returns from subdirectory b to the root directory, and thus file D is accessed.

Detailed Description Text (19):
FIG. 10(A) shows a directory display when the recording mode is set up after the power is turned on subsequent to the loading of a new memory card 14 in the SMC. At this time, no subdirectory has been produced yet, and it is indicated by "01" in the first display section 12a that the pertinent directory is of the first file name 1 in the root directory. By depressing the MD switch 11f for directory production in this state, subdirectory 1 is produced in the first directory of the root directory, and "01" is displayed on the second display section 12b, indicating that the file of file name 1 in the subdirectory 1 is ready for storing (FIG. 10(B)). When the storing is continued up to a limit of file registration in an allowable registration area of the subdirectory, the directory hierarchy is returned to the high-order level, in the instant case the root directory, and "02" is displayed as the next file storage position (FIG. 10(C)).

Detailed Description Text (20):
If it is desired to go to a different directory before the subdirectory file registration limit is reached, the directory position may be ascended, to the root directory for instance, by operating the D-UP switch 11d. The designated file is successively switched by operating the UP and DOWN switches 11b and 11c. When a subdirectory is present, it is accessed, and successive directory designation is effected. When no subdirectory is present, however, the root directory files are successively accessed.

Detailed Description Text (23):
FIG. 11(A) shows the LCD display 12 displaying a hierarchically second level subdirectory registered in the root directory. In this state, "01" indicates a

subdirectory produced in the root directory, and "02" indicates the hierarchically second level directory in that subdirectory. When the D-DOWN switch 11e is operated in this state, the directory position is descended as shown in FIG. 11(B). At this instance, the previous display "02" of the hierarchically second level is shifted to the first display section 12a, and "01" is newly displayed to indicate the first file in a new hierarchically third level subdirectory. At this time, the previous display "01" on the first display section 12a, indicating the designated position of the root directory, vanishes. However, "1" is displayed in the display section 12e to permit confirmation of the presence of one concealed hierarchical level. By subsequently effecting directory return to high-order level by operating the D-UP switch 11d, "01", indicating the subdirectory name in the root directory, is displayed in the first display section 12a, while in the second display section 12b the hierarchically second level subdirectory file access position is incremented by one to display the file name of "03" (FIG. 11(C)).

Detailed Description Text (24):
The plural hierarchical level directory display can be made easier to observe by increasing the number of digits in the display sections.

Detailed Description Text (32):
FIG. 15 is a flow chart of the "last entry routine" as the above subroutine. In this routine, in a first step S41 a check is made as to whether the prevailing storage position is in the root directory. If the position is in the root directory, the routine jumps to a step S45 to generate an alarm, thus bringing an end to this routine. If the position is not in the root directory, the routine goes to a step S42 to read out a high-order level directory pointer for switching over to a hierarchically high-order level directory. In a subsequent step S43 switching over to the high-order level directory is performed, and the file shown by the next directory entry to the switched directory is designated as the file to be stored. Then, in a step S44 the display on the LCD display 12 is updated, thus bringing an end to this routine. While this subroutine was the "UP switch routine" in the recording mode, a similar routine is executed in the playback mode.

Detailed Description Text (33):
FIG. 16 shows the "directory UP routine" as the subroutine noted above. In this routine, in a first step S51 a check is made as to whether the prevailing directory is the root directory. If so, the routine jumps to a step S55 to call a "root directory routine" (see FIG. 18), which is a subroutine to be described later. If the prevailing directory is not the root directory, the routine goes to a step S52 to read out a high-order level directory pointer for switching over to a hierarchically high-order level directory. Then, in a step S54 the display on the LCD display 12 is updated, thus bringing an end to this subroutine. While this subroutine was the "directory UP routine" in the recording mode, a similar routine is executed in the playback mode. The "directory DOWN routine" noted above is similar to the "directory UP routine". In this case, switch-down to low-order level directory is effected.

Detailed Description Text (35):
FIG. 18 is a flow chart of the "root directory routine" as the subroutine noted above. In this routine, in a first step S71 a check is made as to whether the prevailing directory is the last entry. If so, no file can be registered in the pertinent subdirectory, and the routine jumps to a step S74 to generate an alarm, thus bringing an end to this routine. If the prevailing directory is not the last entry, the routine goes to a step S72 to designate the next directory entry as the storage position (storage frame position). Then, in a step S73 the display on the LCD display 12 is updated, thus bringing an end to this routine.

Detailed Description Text (52):
The use of the header section 21 (see FIG. 23) for grouping is shown in detail as group information 27. It is further possible to effect grouping with file names in the "root directory" in memory management area.

Detailed Description Text (53):
When effecting the grouping with the root directory, it is possible to use file names (extender names) and subdirectories.

Detailed Description Text (62):
As in the conventional technique, it is possible to rearrange the sequence of reproduction without shifting actual image data in data file areas. This is done so by changing the directory entry sequence of the root directory in the IC memory card

storage format.

Detailed Description Text (63):
More specifically, where the root directory in the memory management area and its entry format are as shown in FIGS. 30(A) and 30(B), respectively, the reproduction sequence can be managed with the root directory. For example, when the data are stored in the data file areas in the order shown in FIG. 31(B), for reproducing these data in the sequence of C, A, D, E and B, the file names in the root directory may be rearranged to the order as shown in FIG. 31(A).

Detailed Description Text (114):
FIG. 60 shows an example of data organization (file organization) in the storage medium. In the root directory, control file #1, image data files #2 and sound data files #3 are provided. More specifically, the root directory is provided with control file DS00001.J6C, image data files DSC00001.J6I to DSC0004.J6I and sound data files DSC00001.J6S to DSC0003.J6S. It is also provided with subdirectory 01 as "event 1", which is represented as SUB01. The subdirectory is provided with control file #4 and image data files #5. In this example, one control file is provided in each directory for management related to files. In this way, when each directory is copied, the control file thereof is also copied, thus facilitating management of the copy. It is possible to make related management of all the files by providing one control file in the root directory.

Detailed Description Text (117):
Following a start portion START of data area is a frame No. table TABLE. Provided first in the table is root directory #1 of image data. In the root directory, four image data files #2 are provided in correspondence to frame Nos. 1 to 4. Following is sound data file root directory #3, in which three sound data files #4 are provided. Then, control file root directory #5 is provided, in which control file #6 is provided. Then, image data subdirectory #7 is provided, in which eleven image data files #8 are provided. Further, subdirectory #9 is provided, in which control file DCS0101.J6C is provided.

Detailed Description Text (139):
FIG. 69 shows an example of data organization (file organization) in the storage medium. The root directory normally stores control file #1. Following are normally stored three image files and three sound files #2 and #3. Labeled #4 is a continuous high speed storing control file in a continuous high speed storing subdirectory. It can be known from the content of this control file that continuously stored image files have the same organization. Continuously stored image data are accommodated in eleven files #5. In the example of FIG. 69, one control file is provided in each directory for relational management of the files in that directory. However, it is possible to provide a control file in the root directory for relational management of all the files with this control file alone.

Detailed Description Text (146):
FIG. 73 shows an example of the description of the relational information file in the control file of the root directory. The control file in this example includes subdirectory relational information as well.

Detailed Description Text (174):
Referring to the drawing, the root directory has a control file #1 for usual recording and normally stored three image data files and sound data files #2 and #3. In a subdirectory for continuous high speed storing, continuously stored image data are stored in eleven files #4. As shown, a single control file may be provided in the root directory, and relational management of all the files may be made by using this single file alone.

Detailed Description Text (178):
FIG. 90 shows an example of the description of the control file #1 in the root directory shown in FIG. 88.

Detailed Description Text (182):
In a portion #4, a control file of the root directory is described.

Detailed Description Text (262):
The file name that is written in the above way is as shown in FIG. 125. With such a file name it is possible to group individual image data without reading the contents of each file of image data or the like but by merely reading the file name of the root directory. This is effective for improving the retrieval property and also for high

speed reproduction.

Detailed Description Text (289):
As has been shown, in the image data recording apparatus according to the invention a file (control file) showing the relationship of individual data is provided separately of image data for describing information necessary for reproducing all image and speech data files in the file. Thus, when a reproduction instruction is produced, the status of all the files can be known easily without a routine of retrieving the header of a file in question but from the content of the single file. It is thus possible to increase the speed of the process and simplify the file management. More particularly, effective grouping is possible. The apparatus is thus useful for editing, retrieving and efficiency increase of the reproduction. Further, when the control file or file name is referred to, it is possible to obtain grouping without retrieval of all the image data but by merely reading out the file name of in the control file or root directory file. The apparatus is thus suited for high speed processing.

Detailed Description Text (293):
FIG. 141 shows an example of the arrangement of areas in a memory card in conformity to DOS prescriptions. The storage area of the memory card comprises a boot sector area, a FAT area, a root directory area and a data file storage area. The root directory area, as shown in FIG. 142, has continuous areas of directory entries 0, 1, 2, ... In the SMC, frame Nos. 1, 2, . . . are assigned to the entry areas 0, 1. . . . FIG. 143 shows the directory entry. It is divided into areas showing the file name, attribute, reservation, time, date, start cluster and file size. The file name is constituted by 11 bytes, with 8 bytes allocated to the principal file name and 3 bytes to an auxiliary file name. While the following description concerns the root directory, it also applies to the subdirectory.

WEST Search History

DATE: Saturday, April 12, 2003

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L7	l4 and (control$ near access$)	4	L7
L6	l4 and (control near access$)	4	L6
L5	l4 and (control access$)	15	L5
L4	L3 and (directory near level)	15	L4
L3	L1 and (root near directory)	85	L3
L2	L1 (root near directory)	1649	L2
L1	directory near location	698	L1

END OF SEARCH HISTORY

An

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Term	Documents
ACL.USPT.	1566
ACLS.USPT.	215
(3 AND ACL).USPT.	1
(L3 AND ACL).USPT.	1

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L4

Refine Search

Recall Text | Clear

Search History

DATE: Friday, April 11, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L4	L3 and acl	1	L4
L3	L2 and (access$ near right$)	20	L3
L2	L1 and corrielus	360	L2
L1	((707/$)!.CCLS.)	10426	L1

END OF SEARCH HISTORY

WEST

Generate Collection Print

L4: Entry 1 of 15 File: USPT Feb 18, 2003

DOCUMENT-IDENTIFIER: US 6523027 B1
TITLE: Interfacing servers in a Java based e-commerce architecture

Detailed Description Text (1657):
Path: The path follows the package name. If a file is part of the package no. dnb. tb. client, and one is using C: .backslash.Data.backslash.Telebank as your root directory, the path to the file is:

Detailed Description Text (3431):
Additional Tips to Keep in Mind Network users should never be able to execute arbitrary programs or shell commands on your servers--e.g. do not configure your web browser to automatically run spreadsheets or word processors. Because most spreadsheets and word processors these days have the ability to embed executable code within their files. Configure web server so that all CGI scripts or programs must be placed in a single directory. Allow limited access to this directory and it's contents--local users cannot install, change, remove, or edit without review, even prevents them from being read. The practice of allowing any file on web server with .cig extension to be run as CGI script is not recommended. CGI scripts that run on your server must perform either the expected function or return an error message. Scripts should expect and be able to handle any maliciously tailored input. Beware of suste( ), popen( ), pipes, backquotes, an perl's eval( ) function. Avoid spawning subprocesses in CGI scripts and programs. If you must spawn subprocesses, avoid any passing through any strings that are provided by the user. If you must pass strings from the user to the subprocess, be sure that it does not pass shell meta characters. Use a program such as tripwire to monitor for unauthorized changes to the executables and configurations files on your system. Remove the backup files that are automatically generated by your editor. Do not NFS mount or export any web server directories. Delete all compilers on your web server and any utility programs that are not used during boot or by the web server. If possible, place your WWW server and all files in a separate directory structure. Then wrap the WWW server with a small program that does chroot( ) to the directory. Some WWW servers include this approach as an install-time option. If one uses directory level access control files, give them a different name other than the standard .htaccess It is important that the web server password file be inaccessible to normal users on the server and to users over the web. Do not configure the "helper" applications that are automatically run when files of a particular type are downloaded from the net, e.g. provides a way from outsiders to run programs on your computer without your explicit permission, e.g. a program could be embedded in a HTML page as an "included" image. Do not mix HTTP with anonymous FTP. Do not trust the user's browser. HTML includes the ability to display selection lists, limit the length of fields to a certain number of characters, embed hidden data within forms, and specify variables that should be provided to cgi scripts. However, you cannot make your requesting the script's URL; attackers do not need to go through your form or use the interface that you provide. Maintain a good relationship with hardware and software vendors. Monitor vendor communications frequently for discoveries of new bugs or security patches to their software or hardware configurations. If users do not have experience in writing secure programs, it is likely that locally written system. Do not allow users to place scripts or programs on your server unless a qualified application security professional has personally read through the scripts and programs and assured you of their safety. The server's SUID should never be specific as root. If it does, every script that our web server executes may be run as root. However, most servers are designed to be started by the root user, so that it can listen to requests on the standard http TCP/IP port. Do not write SUID root shell scripts or programs on your web server. Server log files record considerable information about each request. Be careful as to how large they grow and check that they automatically trimmed. If they are not monitored for size, log files can file computer's hard disk and result in loss of service. You can learn a lot about the persons who are using the web server. From this information one can get a

comprehensive picture of the people who are accessing the Web, the information that they are viewing, and where they have previously been. Use these logs to monitory the activity occurring on your system.

Detailed Description Paragraph Table (155):
Platform Information Platform Type All Description Migration from Component Test to Assembly Test occurs when the Development team successfully completes the Component Test exit criteria. The timing of the migration should be coordinated between all members of the Development project group. If the project involves more than one platform, the cross platform migration should also be coordinated to be sure that units reach the next phase at the appropriate time. For each platform the migration "kit" should include all units required for the project along with any instructional units. The kit should be created and sent to a staging area until approval for installation in the Assembly Test environment is given. Migration to Stage Information Approval to Stage Development Team Member (the approval must be tracked) (1) Exit/Approval CT Exit Criteria Criteria Kit Creation/Trigger If the Kit creation is automated the trigger should come from Performed by (2) the approver to stage as listed above. & (3) If the Kit creation is not automated, then the Development or TS team should create the kit based upon a portion of the present descriptioned set of procedures. Pre-Migration The pre-migration location for each unit of the migration kit Location should be in an approved library/repository that conforms to the CM repository requirements Post-Migration The post-migration location can be a physically separate Location directory with the appropriate level or security, allowing write access for the kit creation process and read access for the moving of the kits. The post migration location can also be a logical location, where units are tagged with the AT level. Packaging Information Manual/Automated Migration can either be manual or automated. In either case the Package? Tool? process needs to be portion of the present descriptioned and must meet the CM requirements for tracking and recovery. Brief Package Packaging of the CM units should involve a grouping of all Description required units; this grouping should be maintained throughout the entire migration process. This may prevent units from being lost or added during migrations. If multiple units are combined to create a derived product, the creation of the product should be automated by combining like tagged units within the repository (example: a.h, a_sub.pc, a_main.pc, should all be tagged at the CT level). This may prevent the derived product from becoming out of sync with its sub-components in the repository. Only the final product needs to be migrated. Package Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. Verified by The verification should be performed by a Development Project team member, prior to the begirning of Assembly Test, this venfication can be tracked. Internal/External Notification Internal The following teams should be notified upon successful completion of a migration: Development Project Team External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams. Migration from Stage Information Approval from Stage Development Member (the approval must be tracked) (4) Entrance/Approval AT Entrance Criteria Criteria Kit Move Perf by Moving the kit from the staging environment to the installation (5) area can be performed by any person from one of the following teams: Development, Operations, or TS. Movers need to be certain that the appropriate approval has been given prior to moving the kit. Kit Install Perf by Installation of the kit into the new environment can also be (6) performed by multiple groups. Consideration sbould be given to the level of system security access required to perform the installation. Whenever a significant level of access is required, the installation process should be limited to either the TS team or Operations. Pre-Migration The pre-migration location should match the post migration Location location listed above for the Migration to Stage Post-Migration The post-migration location should be a physically separate Location environment from the CT environment whenever feasible and cost effective. This location should mirror the production environment as closely as possible. Un-Packaging/Installation Information Manual/Automated For complex systems and installations requiring a significant Package? Tool? level of access the process should be automated. Manual processes may require explicit directions and a more rigorous verification process. Brief Package Whether the installation process is manual or automated, the Description process should be clearly portion of the present descriptioned. All units should have a specific location on the destination server. Migration from Stage Information The installation process should take into account factors such as space, currently running executables, overwriting existing units, and ?? Install Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. For manual process the verification should be more extensive Verified by The verification should be performed by a Development Project team member, prior to the beginning of Assembly Test. This verification can be tracked. Internal/External Notification Internal The

following teams should be notified upon successful completion of a migration: Development Project Team External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams.

Detailed Description Paragraph Table (156):
Platform Information Platform Type All Description Migration frorn Assembly Test to Product Test occurs when the Development team has successfully completed the Assembly Test exit criteria. The timing of the migration should be coordinated between Development and Test. If the project involves more than one platform, the cross platform migration should also be coordinated to be sure that units reach the next phase at the appropriate time. For each platform the migration "kit" should include all units required for the project along with any instructional units. The kit should be created and sent to a staging area until approval for installation in the Product Test environment is given. Migration to Stage Information Approval to Stage Development Team Member (this approval must be tracked) (1) Exit/Approval AT Exit Criteria Criteria Kit Creation/Trigger If the Kit creation is automated the trigger should come from Performed by (2) the approver to stage as listed above. & (3) If the Kit creation is not automated, then the Development or TS team should created the kit based upon a portion of the present descriptioned set of procedures. Pre-Migration The pre-migration location can be a physically separate Location directory with the appropriate level of security or it can be a logical environment in which the units are tagged with the appropriate migration level. Post-Migration The staging environment can be a physically separate directory Location with the appropriate level of security or it can be a logically separate environment in which the units are tagged with the appropriate migration level. Packaging Information Manual/Automated Migration can either be manual or automated. In either case the Package? Tool? process needs to be portion of the present descriptioned and must meet the CM requirements for tracking and recovery. Brief Package Packaging of the CM units should involve utilizing the same Description grouping as the migration from CT to AT, this may prevent the introduction of new units or the loss of required units. Migration to Stage Information If multiple units are combined to create a derived product then only the derived product needs to be migrated. Some environments may require the product to be created differently for each destination environment, in this case the sub- components need to be migrated as well. Package Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. This procedure may require an extra step during the actual packaging to create an audit log identifying the status of the migration. Verified by The verification should be performed by a Development Project team member, prior to notifying Test. Internal/External Notification Internal The following teams should be notified upon successful completion of a migration: Test Team External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams. Migration from Stage Information Approval from Stage Development Member (this approval must to be tracked) (4) Entrance/Approval PT Entrance Criteria Criteria Kit Move Perf by Moving the kit ftom the staging environment to the installation (5) area can be performed by any person from one of the following teams: Development, Operations, or TS. Movers need to be certain that the appropriate approval has been given prior to moving the kit. Kit Install Perf by Installation of the kit into the new environment can also be (6) performed by multiple groups. Consideration should be given to the level of system security access required to perform the installation. Whenever a significant level of access is required, the installation process should be limited to either the TS team or Operations. Pre-Migration The pre-migration location should match the post migration Location location listed above for the Migration to Stage Post-Migration The post-migration location should be a physically separate Location environment from the CT environment whenever feasible and cost effective. This location should mirror the production environment as closely as possible. Un-Packaging/Installation Information Manual/Automated For complex systems and installations requiring a significant Package? Tool? level of access the process should be automated. Manual process may require explicit directions and a more rigorous verification process. Brief Package Whether the installation process is manual or automated, the Description process should be clearly portion of the present descriptioned. All units should have a specific location on the destination server. The installation process should take into account factors such as space, currently running executables, overwriting existing units, and?? Install Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. This procedure may require an extra step during the actual packaging to create an audit log identifying the status of the migration. Migration from Stage Information For manual processes the verification should be more extensive Verified by The verification should be performed by an Test member, prior to

the beginning of Product Test. Internal/External Notification Internal The following teams should be notified upon successful completion of the migration: Test External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams.

WEST

Generate Collection Print

L4: Entry 2 of 15 File: USPT Dec 17, 2002

DOCUMENT-IDENTIFIER: US 6496944 B1
TITLE: Method for database assisted file system restore

Detailed Description Text (16):
INODE 104 is an identifier internal to a particular filesystem which keeps track of the location of a directory. When a filesystem is first created under operating system control, one of the areas allocated to management information rather than data storage is an INODE list. The INODE list contains an array of INODEs which themselves contain important information about an associated file. File ownership, file access permissions and disk address are some examples of information in an INODE. Furthermore, an INODE is referred to by its INODE number--where in the INODE list it occurs. The INODE number of the root directory of a filesystem is assumed to be 0 in the current invention. However, referring to root directories using numbers other than 0 are also contemplated to be within the scope of the present invention. In each filesystem, the operating system maintains a directory information table which relates directory entries (e.g. files, directories, symbolic links) with INODE numbers.

Detailed Description Text (26):
Step 1. Locate all the root-level directories by searching for all entries in the table with INODE value equal to 0 (zero). For each such located directory re-create them in the filesystem and perform steps 2 through 5. An index scan of the table's primary key is one efficient method of locating these entries.

Detailed Description Text (32):
The algorithm identified above is illustrated in the flowchart of FIG. 2: When an entire filesystem restore is started 200, the first step 202 locates a root level directory entry by searching the database table for entries having INODE values equal to 0. If no root level directory is located, then the filesystem restore is complete at step 204. When, however, a root level directory entry is located then a directory is re-created, step 206, in the filesystem for that entry. The next step, 208, extracts the FSID and INODE values from the directory entry located in step 202 and defines a current directory by setting the values c_inode and c_fsid equal to the extracted INODE and the extracted FSID values, respectively.

Detailed Description Text (34):
With these new current directory values, step 228 returns to step 210 to continue. If an entry corresponding to a child directory of the current directory is not located in step 210, then step 212 determines whether the current directory is a root level directory. If the current directory is a root level directory, then step 214 returns to step 202 which locates the next root level directory entry to process. If the current directory is not a root level directory, then step 216 changes the current directory by restoring c_fsid and c_inode to the values previously stored in step 224. With these new current directory values, step 218 returns to step 210 to continue.

Detailed Description Text (35):
Performing the above algorithm reconstructs an entire file system top down from the table by first finding the root-level directories for each filesystem and then recursively finding its children directories and its children's children directories, etc. as illustrated in the sequence of FIGS. 3a through 3g which detail the progression of the directory structure rebuilding according to the table entries in FIG. 1.

Detailed Description Text (45):
In step 408, if the extracted Parent INODE value is NULL (i.e. a root level directory) then upwardly traversing the subdirectory path is finished and control transfers to step 410. Step 410 rebuilds the desired subdirectory in the filesystem by re-creating the subdirectory's path according to all the extracted Directory Names in a last-found,

first-out order.

CLAIMS:

10. A method to restore a filesystem's directory structure to any point in time, as per claim 1, wherein said reconstructing said filesystem further comprises the steps: i. identifying at least one root-level directory entry and re-creating said directory in said filesystem; ii. recursively identifying all children directory entries of said root-level directory and recreating said children directories in said filesystem; iii. repeating step ii until all descendant directories of said root-level directory are re-created, and iv. repeating steps ii and iii for each of said root-level directory entries identified in step i.

11. A method to restore a filesystem's directory structure to any point in time, as per claim 2, wherein said reconstructing said filesystem further comprises the steps: A. identifying each of said table entries which correspond to a root-level directory in said filesystem; B. re-creating each of said directories identified in step A; C. for each of said root-level directory entries identified in step A, performing the following steps: i. storing said root-level directory entry's filesystem identifier in a variable, current_fsid; and storing said root-level directory entry's node identifier in a variable, current_inode; ii. identifying each table entry which satisfies both: a. said table entry's parent node identifier equals current_inode, and b. said table entry's filesystem identifier equals current_fsid; iii. re-creating a directory/subdirectory for said table entry identified in step ii; iv. for each of said table entries identified in step ii, storing said entry's filesystem identifier in a variable, current_fsid; and storing said entry's node identifier in a variable, current_inode, and v. recursively applying steps ii through iv until no matching table entries are identified in step ii.

WEST

Generate Collection Print

L4: Entry 4 of 15 File: USPT Apr 30, 2002

DOCUMENT-IDENTIFIER: US 6381615 B2
TITLE: Method and apparatus for translating virtual path file access operations to physical file path access

Brief Summary Text (8):
Two other mechanisms common in Unix.RTM. operating systems are symbolic links and hard links. Unix.RTM. is a trademark of the Open Group. In the Unix.RTM. operating system, each file and directory location is defined by an "ionode" number, and the file system directory maintains mappings between file and directory names and ionode numbers. A hard link is simply a second mapping between a file or directory name and an ionode number that has already been mapped to a first name. The file or directory is not actually deleted until all file and directory names that map to the ionode number have been deleted. Hard links must reference files and directories on the same file system, are transparent to the application, and cannot be used to arbitrarily re-map between partitions or devices because the ionode numbers only refer to files and directories within the file system.

Brief Summary Text (10):
Mount points can also virtualize file systems to some extent. Amount point simply allows a drive volume to be mounted at some point within an existing directory tree structure. For example, assume that a hard drive on a file sever has a root directory, and under the root directory are a series of directories, with each directory identified by the initials of the user that uses that directory. When a user logs in, a drive volume is mounted at the directory corresponding to the users initials. When the user accesses that drive volume, the user only sees directories and files beneath the mounting point, and does not see the directories of the other users. Mount points are transparent to applications and can be re-mapped arbitrarily, but do not provide file level granularity and do not re-map directories underneath the mount point.

Brief Summary Text (11):
Finally, some enterprise storage subsystems have the ability to move a user's file system from one hard drive to another. Such moves can be performed arbitrarily and are transparent to applications, but do not provide file or directory level granularity.

WEST

Generate Collection Print

L4: Entry 5 of 15 File: USPT Feb 26, 2002

DOCUMENT-IDENTIFIER: US 6351741 B1
TITLE: Method of locating a file linked to a document in a relocated document directory structure

Brief Summary Text (3):
One common way to organize files is with a directory structure that allows files or directories to reside within other directories. When a directory resides within another directory, the residing directory is referred to as a "subdirectory". A file ultimately resides in a parent directory. The file's parent directory may, in turn, have a parent directory, and so on, thus creating a hierarchy of directories. To access a particular file, each of the parent directories in a chain from a starting or "root," directory are named in a string, along with the file's name. The resulting string is referred to as a "pathname".

Brief Summary Text (5):
Consider as an example a file "fileX" with an absolute pathname "HardDrive: .backslash.directoryA.backslash.directoryB.backslash.directoryC.backslash. fileX". The directory "directoryA" is the root directory, which is always given as a starting directory in an absolute pathname.

Brief Summary Text (6):
Another form of pathname is a relative pathname, which identifies a file by using a relative path and a reference point such as a "current" or starting directory (other than the root directory). Using the same example as above, and assuming the current directory is "directory C", a relative path to be used to access "fileX" is ".backslash.directoryC.backslash.fileX".

Detailed Description Text (3):
Referring to FIG. 2, the step or process of constructing a new absolute pathname (step 16 of FIG. 1) is shown. The process of constructing a new absolute pathname 16 parses the absolute pathname of the linked file into portions corresponding to a filename or directories (step 20). It begins the parsing with a first portion corresponding to the filename of the linked file and works towards a last portion corresponding to the root directory. The first portion or filename corresponds to a lowest level relative pathname. The first portion taken in conjunction with the next parsed portion corresponds to a next higher level relative pathname and so on. Taking the lowest level relative pathname as the "current" relative pathname (step 21), the new absolute pathname constructing process appends the current relative pathname to the directory specification (i.e., location) of the document to produce a new absolute pathname (step 22). The process 16 determines if the resulting new absolute pathname points to the file (step 24). If the process does not find the file, it proceeds to a next higher level relative pathname and repeats the process of appending the next higher level relative pathname to the directory specification of the document (step 22) and determining if the new absolute pathname is the location at which the file is stored (step 24). It continues proceeding to a next higher-level relative pathname until the new absolute pathname corresponds to the file location. The process 16 replaces the absolute pathname with the new absolute pathname (step 26).

Detailed Description Text (9):
If, subsequent to storing `MYDOCUMENT.INDD` in the `PUBS` directory on the C volume (or drive), the document and its associated subdirectories and linked files--including `MYIMAGE.TIF`--are moved to a different location, such as a different disk, and that the top-level directory `PUBS` has been renamed `FINISHED.backslash.DOCUMENTS`. Thus, the absolute pathname for the document MYDOCUMENT.INDD is now `D:.backslash.FINISHED.backslash.DOCUMENTS.backslash.MYDOCUMENT.INDD`. The absolute pathname for the linked file is `D:

.backslash.FINISHED.backslash.DOCUMENTS.backslash.IMAGES.backslash.TIFF.ba
ckslash.MYIMAGE.TIF`. If an application opens the `MYDOCUMENT.INDD` document, the application attempts to locate the image file by following the recorded absolute pathname, i.e., the original absolute pathname `C:.backslash.PUBS.backslash.IMAGES.backslash.TIFF.backslash.MYIMAGE.TIF`. It determines that the recorded or specified pathname is no longer valid. Using the new absolute pathname constructing process 16 as described above with reference to FIG. 2, the application can construct a new absolute pathname pointing to the linked file's new location.

Detailed Description Text (11):
Referring to FIG. 3, an old absolute pathname 30 for `MYDOCUMENT.DOC` at an old location 32 and a new absolute pathname 34 for `MYDOCUMENT.INDD` at a new location or directory specification `D:.backslash.FINISHED.backslash.DOCUMENTS` 36 is shown. Also shown is an old absolute pathname 38 for a linked file `MYIMAGE.TIF`. An application appends to the new directory specification 36 a filename `MYIMAGE.TIF` 40, which it obtains by parsing the original absolute pathname 38. The application checks for the file having the filename `MYIMAGE.TIF` at the location corresponding to the resulting new absolute pathname `D:.backslash.FINISHED.backslash.DOCUMENTS.backslash.MYIMAGE.TIF` 42. If the file is not located there, the application appends to the directory specification 36 the filename `MYIMAGE.TIF` 40 combined with a next higher level directory `TIFF` 44 from the original (stored) absolute pathname, that is, a next higher level relative pathname `TIFF.backslash.MYIMAGE.TIF` 46. Again, the application checks for the file in the location corresponding to the resulting new absolute pathname `D: .backslash.FINISHED.backslash.DOCUMENTS.backslash.TIFF.backslash.MYIMAGE.
TIF` 48. If the application determines that the file is not stored at this second location, it takes a next higher level directory `IMAGES` 50 in conjunction with `TIFF.backslash.MYIMAGE.TIF.backslash.` 46 as a next higher level relative pathname `IMAGES.backslash.TIFF.backslash.MYIMAGE.TIF` 52 and appends it to the directory specification D:.backslash.FINISHED.backslash.DOCUMENTS.backslash. 36 to give a new absolute pathname `D:
.backslash.FINISHED.backslash.DOCUMENTS.backslash.IMAGES.backslash.TIFF.ba
ckslash.MYIMAGE.TIF` 54. The application follows this pathname to locate the "missing" linked file `MYIMAGE.TIF`.

WEST

Generate Collection Print

L4: Entry 6 of 15 File: USPT Jul 3, 2001

DOCUMENT-IDENTIFIER: US 6256031 B1
TITLE: Integration of physical and virtual namespace

Brief Summary Text (4):
Developers of Web sites or FTP (file transfer protocol) sites, particularly complex sites, first need to determine how the various files of that site are to be organized. Then, appropriate directories are created according to the planned organization, for example, separate directories are created for storing HTML (Hypertext Markup Language) pages and for storing program files, e.g., Active Server Pages (ASP) applications or custom programs. Once created, the directories containing the documents that are to be published are specified to a management service or the like. One of the directories is a home directory, which is the root directory for a site wherein content files are stored. The home directory is accessible to the user and contains files and programs, and typically contains the home page, i.e., the initial page of information for a collection of pages. In other words, the home directory is the central location for published pages, and often includes a home page or index file that welcomes users and contains links to other pages in the site.

Brief Summary Text (7):
Network administrators, however, have heretofore always had to deal with the actual locations of the directories. This is because administrative tools are device-oriented rather than site-oriented. For example, to change a property on a file under the home directory, the administrator would use a local user interface, while to change a property on a remote file, the administrator would use a user interface for remote operation, such as a network browser. In other words, in existing site management applications and tools, the physical and virtual namespaces are displayed as different entities. Further, existing management tools have not provided any indication of the site's hierarchy, whereby to manage a site, the administrator must work through various mappings in order to locate the various directories and files therein. As a result, if there is some relationship between properties of files, the administrator is unable to take advantage of this relationship. In particular, when files are stored at different physical locations, the administrator must not only work with the individual files, but also must do so via the distinct types of management tools.

Detailed Description Text (21):
Keys directly below the server keys correspond to root virtual directories, subordinate virtual directories, disk directories, and information objects. Virtual directories have no direct correlation in the physical storage media, while physical directories have a direct counterpart in the physical storage media. In some situations, virtual root directories may be mapped to a physical counterpart that may not be a root directory. Other keys illustrated in FIG. 4 also correspond to various objects of the information server installation.

Detailed Description Text (26):
As described above, the content of a site may be organized into virtual directories and physical directories. Virtual directories are those that are not within the home directory, and which have an alias, a name that client browsers use to access that directory. Aliases provide a number of benefits, including making it easier for administrators to move directories in a site. More particularly, instead of changing the URL for the directory, to move a directory, the mapping between the alias and the physical location of the directory is changed. Also, because an alias is usually shorter than the path name of the directory, it is more convenient for users to type. Moreover, an alias is more secure, because with an alias, users do not know where files are physically located on the server, and thus cannot use that information to modify the files.

Detailed Description Text (28):
Note that the physical directories may be considered as not having an alias, or alternatively, as having as their alias their precise pathname. In any event, as can be readily appreciated, /SiteAdmin and /PR are aliases for virtual directories actually located at .backslash..backslash.Server2.backslash.Scripts.backslash.Admin and D:.backslash.Mktng.backslash.PR, respectively. Note that the client views the site via its URL path, and thus can see the hierarchical organization of the site, but has no idea of the actual locations of the directories.

Detailed Description Text (35):
In keeping with the invention, the properties maintained on different physical locations may now benefit from the concept of property inheritance. With inheritance, (also described in the aforementioned U.S. patent application Ser. No. 08/963,894), the property set for a parent node is inheritable by all child nodes. For example, during the installation of Internet Information Server, default values were assigned to the various properties on the property sheets. Properties may thereafter be set on the site level, directory level, and even on the file level. Settings on higher levels (such as the site level) are automatically passed on, or inherited, by the lower levels (such as the directory level) but can still be individually edited at the lower level. Once a property has been changed on an individual site, directory, or file, changes to the master defaults will not automatically override the individual setting.